6502 carry flag cheat sheet

v1.0

Carry-related instructions

instructions that use C:
BCC: branch if C is 0
BCS: branch if C is 1
BRK/PHP: pushes C onto stack

instructions that modify C:
CLC: C=0
SEC: C=1
CMP/CPX/CPY:
   if reg < mem then C=0 else C=1
ASL: $C=M_7$, $M_7=M_6$, ..., $M_1=M_0$, $M_0=0$
LSR: $M_7=0$, $M_6=M_7$, ..., $M_0=M_1$, $C=M_0$
PLP/RTI: pulls C from stack

instructions that use then modify C:
ADC: if A+M+C > $FF then C=1 else C=0
SBC: if A-M+C-1 < $00 then C=0 else C=1
ROL: $M_7=M_6$, ..., $M_1=M_0$, M_0=C then $C=M_7$
ROR: M_7=C, $M_6=M_7$, ..., $M_0=M_1$ then $C=M_0$

Arithmetic

ADC
- before ADC, perform CLC
- if C known, can skip CLC
    (but if C is 1, ADC #imm-1)
- if overflow (result > $FF
    or result > $99 in BCD mode)
      then C=1 else C=0

SBC
- before SBC, perform SEC
- if C known, can skip SEC
    (but if C is 0, SBC #imm+1)
- if underflow (result < $00)
    then C=0 else C=1

16-bit * 2:
```
ASL num_lo
ROL num_hi
```

int(16-bit / 2):
```
LSR num_hi
ROR num_lo
```

signed int(8-bit / 2):
```
CMP #$80
ROR
```

toggle carry:
```
ROL
EOR #$01
ROR
```

Unsigned comparisons

8-bit compare: (CMP/CPX/CPY)
- if reg < mem then C=0
- if mem > reg then C=0
- if reg >= mem then C=1
- if mem <= reg then C=1

16-bit check if num1 < num2:
```
        ; (using acc, x, or y)
        LDA num1_hi
        CMP num2_hi
        BCC less
        BNE not_less
        LDA num1_lo
        CMP num2_lo
        BCC less
not_less ...
```

16-bit check if num1 < num2:
```
        ; (using acc)
        LDA num1_lo
        CMP num2_lo
        LDA num1_hi
        SBC num2_hi
        BCC less
not_less ...
```

equality check without C:
```
   EOR #const
   BEQ is_equal
```

16-bit + 8-bit:
```
LDA num_lo
CLC
ADC #const
STA num_lo
BCC after_inc
INC num_hi
```

16-bit - 8bit:
```
LDA num_lo
SEC
SBC #const
STA num_lo
BCS after_dec
DEC num_hi
```

16-bit + 16-bit:
```
CLC
LDA num1_lo
ADC num2_lo
STA sum_lo
LDA num1_hi
ADC num2_hi
STA sum_hi
```

16-bit – 16-bit:
```
SEC
LDA num1_lo
SBC num2_lo
STA diff_lo
LDA num1_hi
SBC num2_hi
STA diff_hi
```